EXPRESS JS / AJAX

Michael Gfeller

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

Inhaltsverzeichnis

https://github.com/gfeller/Vorlesung_Express

Lernziele

Die Teilnehmer…

- **… können die einzelnen Komponenten / Module einer Express Applikation anwenden**
- **… finden sich in einer Express-Applikation zurecht**
- **… sind in der Lage eine eigene Express-Applikation zu entwickeln**
- **… sind in der Lage nedb als non-sql Datenbank zu nutzen**
- **… können Callbacks korrekt anwenden**
- **… kennen die Gemeinsamkeiten / Unterschiede von Cookie und Session**
- **… können Daten von einer REST-Schnittstelle nachladen**
- **… kennen das Konzept von Sockets**

JSON

JSON (JavaScript Object Notation)

- **JSON ist ein Daten-Austauschformat.**
- **Wird verwendet um Daten zu senden, speichern (z.B. DOM storages, cookies, HTTP requests).**
- **Hat im Web XML verdrängt**
- **Wird oft mit AJAX verwendet**
- **Unterschied zwischen JSON und JavaScript Object**
  - all identifiers and all strings are written between double quotes
  - the only allowed data types are: String, Number, Boolean, Array, Object and null.
    - NaN, Infinity und -Infinity are converted to null
  - there is no defined representation for Date, Error, Regular Expression, and Function objects.
  - there are no comments
  - trailing commas are forbidden
- **Content-Type: application/json**
- **JSON-Helper: JSON.parse & JSON.stringify**

JSON - Array

- **JSON - Array ist Valid sollte aber nicht verwendet da:**
  - **JSON Hijacking**
    - https://cheatsheetseries.owasp.org/cheatsheets/AJAX_Security_Cheat_Sheet.html#protect-against-json-hijacking-for-older-browsers
  - **Erweiterbarkeit**

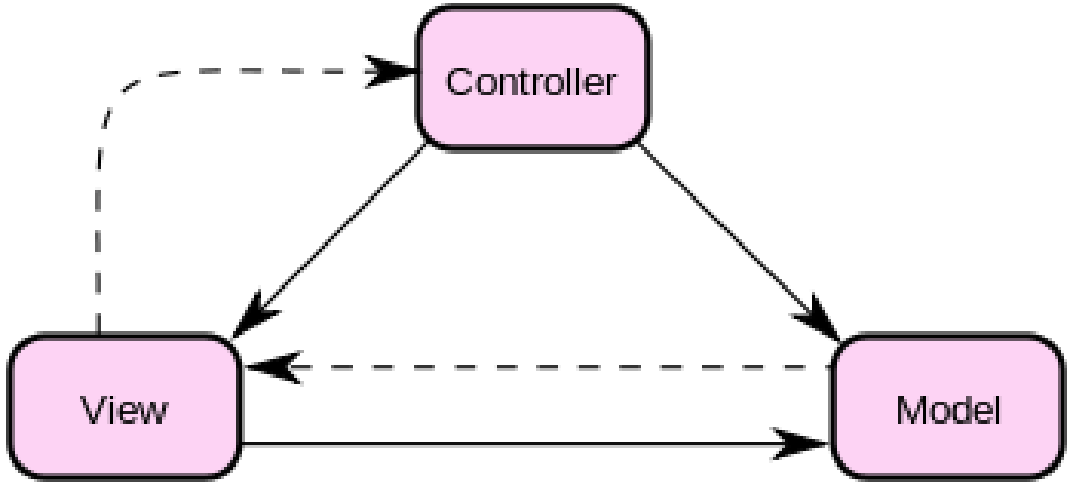
```
//Gültig aber nicht ideal & gefährlich:
JSON.stringify([1,2,4,5]);

//besser:
JSON.stringify({elements : [1,2,4,5]});
```

- **Validator: http://jsonlint.com/**

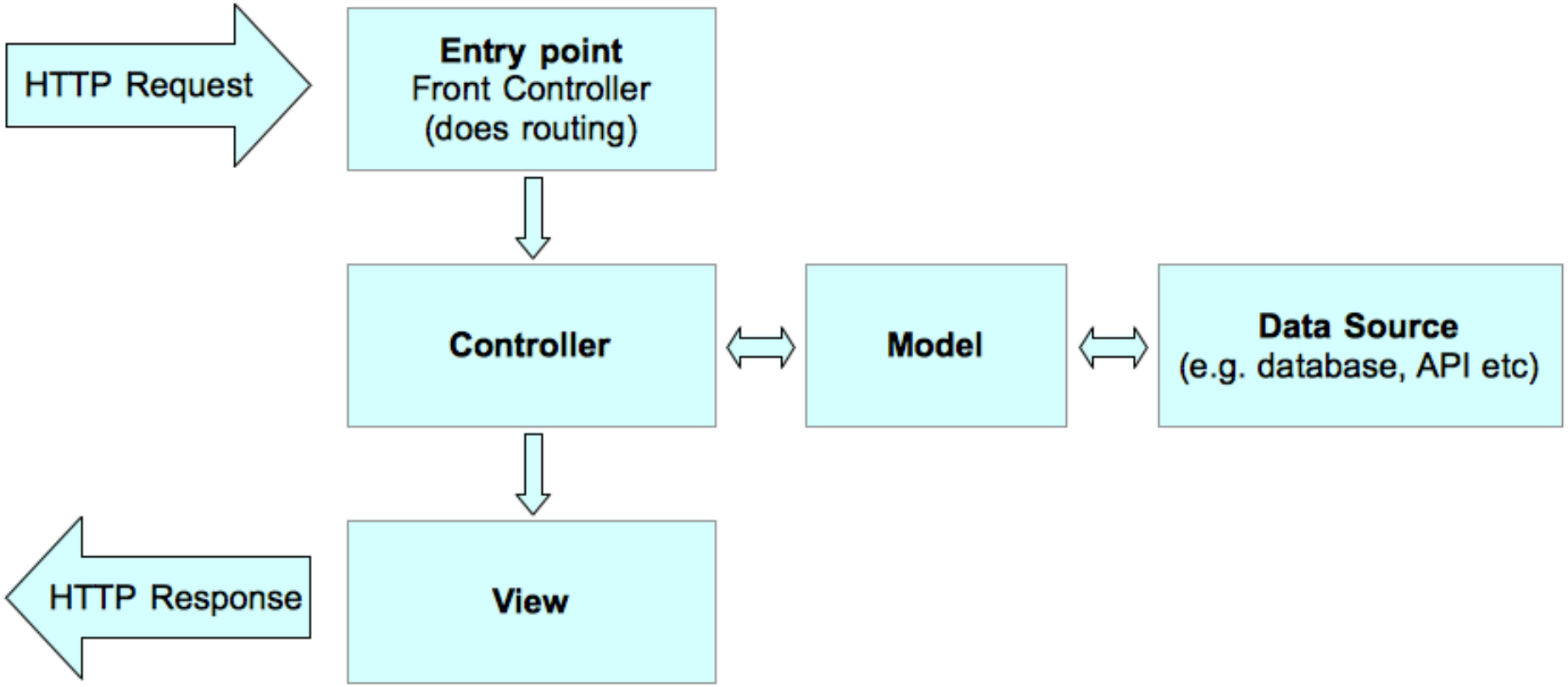
PATTERNS

MVC-Pattern

- **Model: Daten und Datenaufbereitung**
- **Controller**
  - Verknüpft die View mit den Daten
- **View**
  - Darstellen der Daten



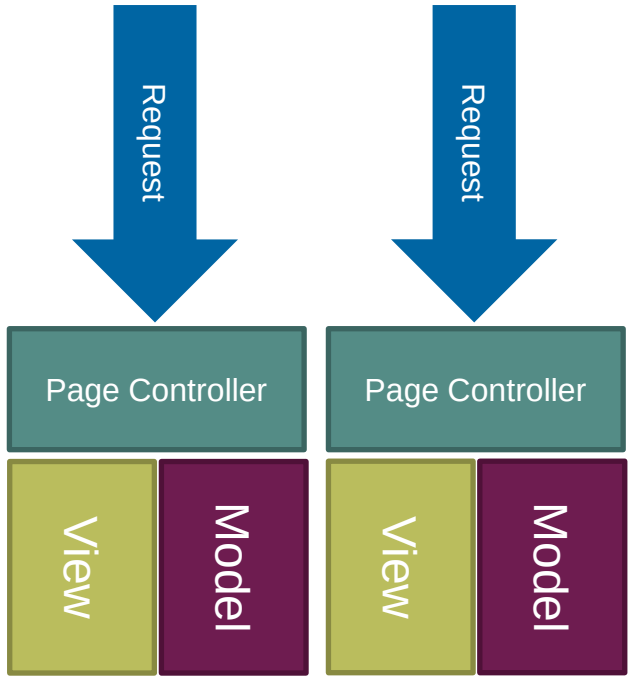
Front Controller



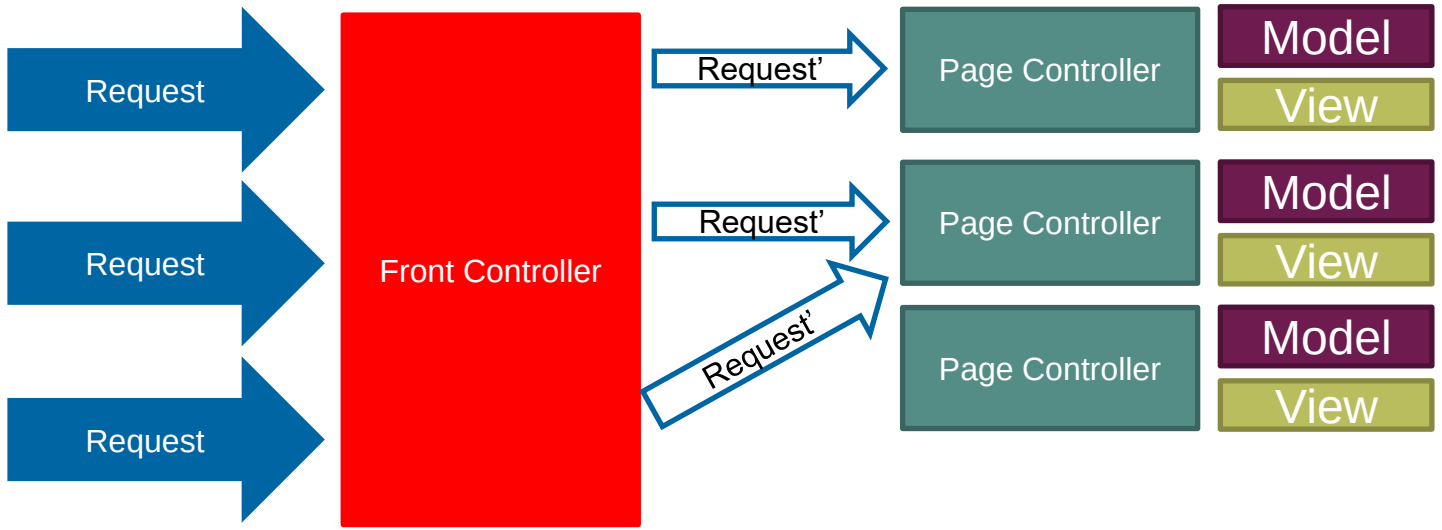
http://martinfowler.com/eaaCatalog/frontController.html

Page Controller vs. Front Controller

Page Controller



Front Controller



EXPRESS

Warum Express?

- **Meist genutztes Web Framework**
  - Etwas in die Jahre gekommen
  - https://www.npmjs.com/package/express

- **Alternativen**
  - koa: https://www.npmjs.com/package/koa
  - hapi: https://www.npmjs.com/package/@hapi/hapi
  - restify: https://www.npmjs.com/package/restify
  - …

Installation

- **npm install express**
  - npm i express
- **Spezifische Version kann wie folgt installiert werden**
  - npm i express@4.16.3

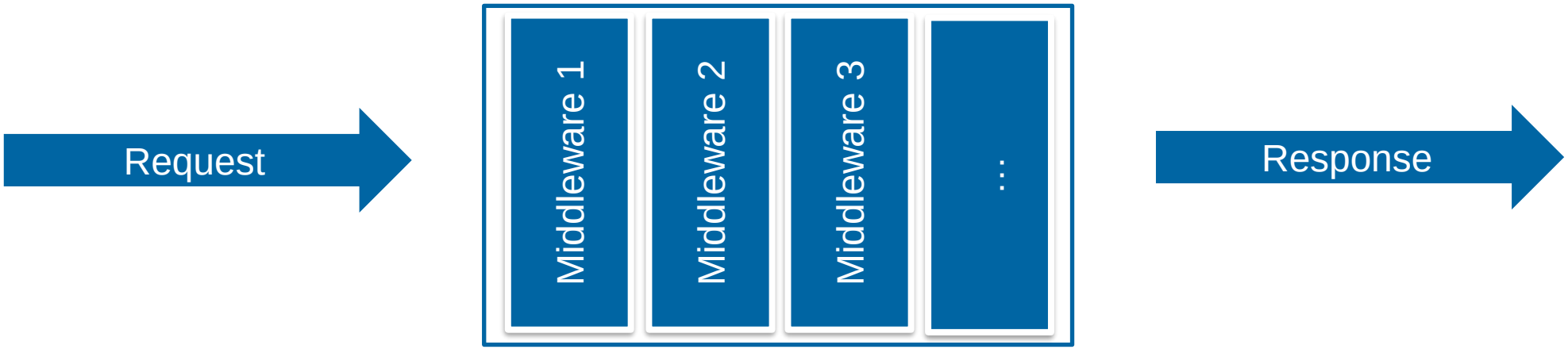
- **Nutzen wie folgt:**

```
const http = require('http');
const express = require('express');
const app = express();

http.createServer(app).listen(3000);
```

```
const express = require('express');
const app = express();

app.listen(3000, function () {
    console.log('Example app listening on port 3000!');
});
```

Middleware

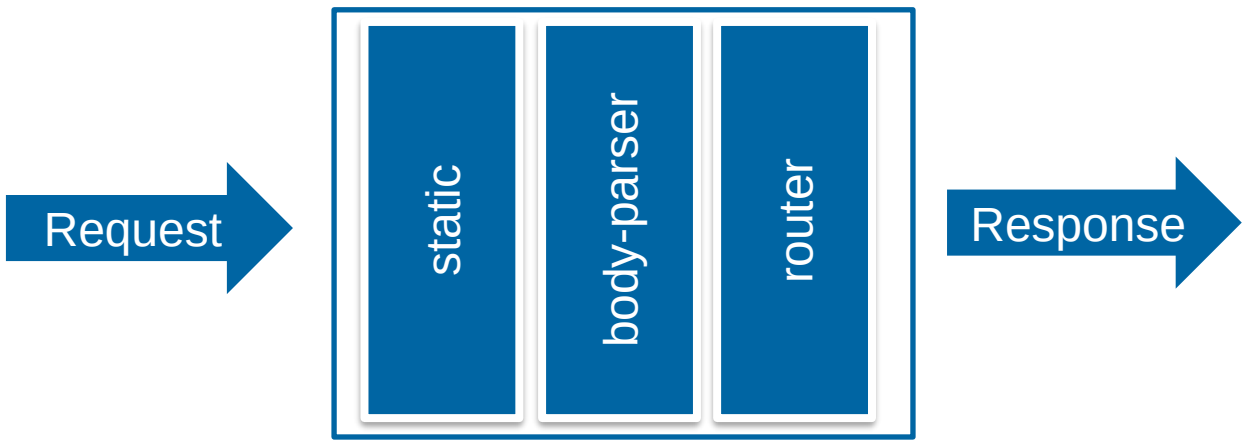
- **Express nutzt Middleware für die Request Bearbeitung**
- **Middleware ist ein Stack von Anweisungen welche für ein Request ausgeführt wird.**



Middleware registrieren

- **Mit app.use(…) wird eine neue Middleware registriert**
- **Die Reihenfolge der Registrierung bestimmt die Ausführungsreihenfolge**

```
const express = require('express');
const bodyParser = require('body-parser');

const app = express();
const router = express.Router();

app.use(express.static(__dirname + '/public'));
app.use(bodyParser.urlencoded({ extended: false }));
app.use(router);
```



Middleware-Sammlung: Connect

- **Wurde bis zu V3 von Express direkt genutzt und definierte die Middleware Logik**
- **Ab V4 keine direkte Verbindung und eigene Middleware Implementation.**
  Die Middlewares sind kompatible.
- **Connect definiert viele nützliche Middlewares welche von Express.js-Applikation genutzt werden können.**
- **https://github.com/senchalabs/connect#middleware**
  - body-parser
  - cookie-parser
  - cookie-session
  - errorhandler
  - method-override
  - serve-static
  - …

Express

- **Middlewares**
  - Router
  - Static
- **Erweitert Request: http://expressjs.com/4x/api.html#req**
  - params
  - is, get
- **Erweitert Response**
  - sendFile
  - Cookie
  - format
  - json / jsonp
- **Server-Side-Rendering**

DEMO 1 - ROUTING

Router-Middleware

- **Middleware befindet sich auf dem Express Objekt**

```
const express = require('express');
const router = express.Router;
```

- **Wichtige Methoden**
  - **router.all(path, [callback, ...] callback)**
    - Wird unabhängig vom der HTTP Methode aufgerufen
  - **router.METHOD(path, [callback, ...] callback)** (METHOD : get, put, post, delete)
    - Wird aufgerufen, falls die jeweilige HTTP Methode verwendet wurde

```
router.get('/', function(req, res){
    res.send('hello world');
});
```

  - **router.route(path)**
    - Kann benutzt werden um für einen Path verschiedene Methoden zu gruppieren

```
app.route('/book')
    .get(function(req, res) {res.send('Get a random book');})
    .post(function(req, res) {res.send('Add a book');})
```

- **All diese Methoden liegen auch direkt auf der Express-Applikation.**

Router-Middleware

- **router.all(path, [callback, ...] callback)**
  - Verwendet Pattern matching. Beispiele:
    - '/ab*cd‘
    - ‘/*’
  - Dynamische Werte
    - ‘/orders/:id/’
    - Der Wert, welcher in :id geparsed wurde, wird in req.params.id abgespeichert
  - Es können mehrere Callbacks als Chain übergeben werden.

Router-Middleware

- **router.all(path, [callback, ...] callback)**
  - Verwendet Pattern matching. Beispiele:
    - '/ab*cd‘
    - ‘/*’
  - Dynamische Werte
    - ‘/orders/:id/’
    - Der Wert, welcher in :id geparsed wurde, wird in req.params.id abgespeichert
  - Es können mehrere Callbacks als Chain übergeben werden.

Static-Middleware

- **Aufgabe: Statische Files ausliefern**
- **Nutzen wie folgt:**
  - ***app***.use(***express***.**static**(__dirname + **'/public'**))
- **Es sind mehrere static-routes möglich.**

Custom-Middleware

- **Hat 3 Parameter**
  - request
  - response
  - next
    - Zeigt auf die nächste Middleware im Stack
- **Next kann aufgerufen werden, um die nächste Middleware aufzurufen.**
  - Dies kann auch unterlassen werden. In diesem Falle sollte die Anfrage beantwortet werden.

```
function myDummyLogger( options ){
    options = options ? options : {};

    return function myInnerDummyLogger(req, res, next)
    {
        console.log(req.method +":"+ req.url);
        next();
    }
}
```

Error-Middleware

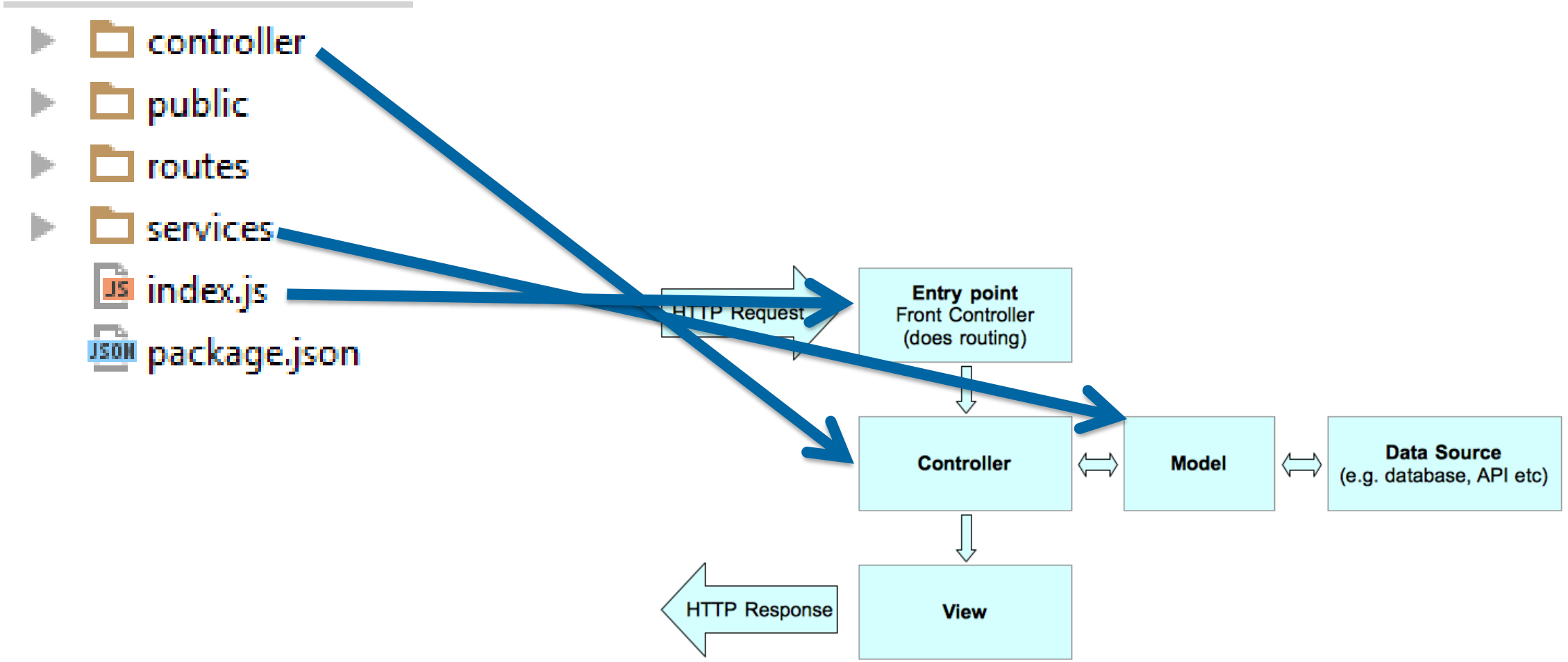
- **Aufgabe: Bearbeitet Errors, welche von den Middlewares generiert wurden**
- **Details: http://expressjs.com/guide/error-handling.html**
- **Die Error-Middleware muss 4 Parameter haben**
  - error
  - request
  - Response
  - Next

```
app.use(function(err, req, res, next) {
  console.error(err.stack);
  res.status(500).send('Something broke!');
});
```

- **Die Error-Middleware sollte als die letzte Middleware registriert werden.**
- **Es können mehrere Error-Middlewares registriert werden.**
  - Die letzte Error-Middleware muss die Anfrage beenden.
- **Wird aufgerufen, falls ein Error-Objekt dem Next-Callback übergeben wird.**
  - Ab diesem Zeitpunkt werden keine normalen Middlewares mehr aufgerufen.

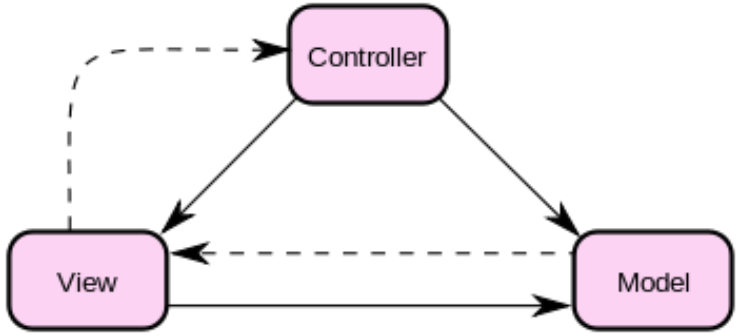
DEMO 2 / 3 - MODEL

Struktur


controller
public
routes
services
index.js
package.json
HTTP Request
Entry point
Front Controller
(does routing)
Controller
Model
Data Source
(e.g. database, API etc)
View
HTTP Response

Model

- **Ziel: Die Daten sollten in einem Module verwaltet und abgespeichert werden.**
- **Möglichkeiten um Daten zu speichern:**
  - In Memory: Array
  - JSON
  - NoSQL-Datenbanken
    - https://www.mongodb.org/
    - https://github.com/louischatriot/nedb
  - Sql-Datenbanken
    - https://github.com/mapbox/node-sqlite3
  - Oracle-Datenbank
    - https://www.npmjs.com/package/db-oracle
  - …



NoSQL-Datenbanken am Beispiel nedb

- **NoSQL Datenbanken sind Dokumenten-basierend.**
  - Jedes Dokument beinhaltet alle Daten, welche notwendig sind.
  - Relationen können über «keys» manuell erstellt werden
    - Relationale-Integrität muss die Applikation selbst sicher stellen oder zumindest damit umgehen können!
- **Beispiel laden der Datenbank**

```
const Datastore = require('nedb');
const db = new Datastore({ filename: './data/order.db', autoload: true });
```

- **Einfügen**
  - Beim Einfügen wird ein Feld _id gesetzt, welches die eindeutige ID von der Datenbank beinhaltet

```
db.insert(order, function(err, newDoc){
    if(callback){
        callback(err, newDoc);
    }
});
```

NoSQL-Datenbanken am Beispiel nedb

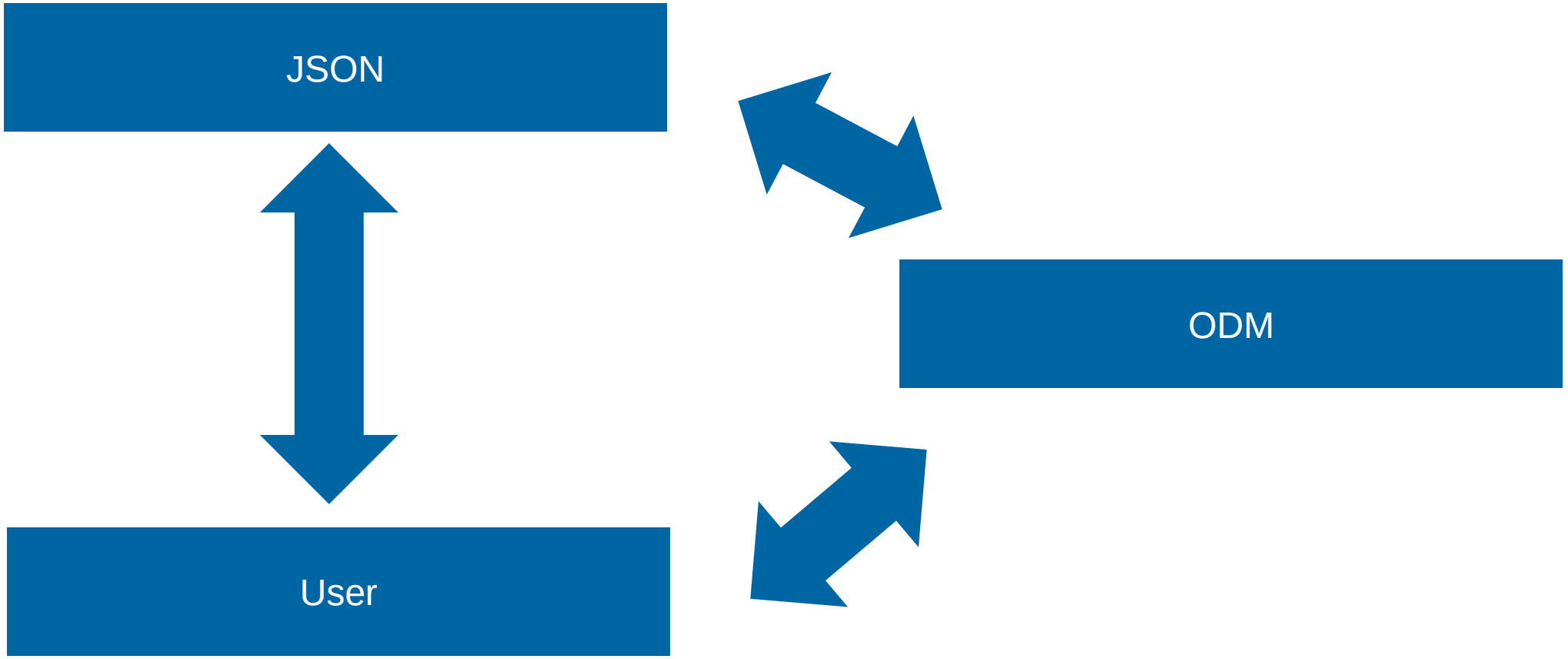
- **Suchen**
  - find oder findAll

```
db.findOne({ _id: id }, function (err, doc) {
        callback( err, doc);
    });
```

- **Updaten**
  - Viele Möglichkeiten:
    - Einzelne Werte ändern
    - Array von einem «document» anpassen
    - Ganzes Objekt ersetzten
  - https://github.com/louischatriot/nedb#updating-documents

```
db.update({_id: id}, {$set: {"state": "DELETED"}}, {}, function (err, doc) {
    publicGet(id,callback);
});
```

ODM (object document mapping) mit Mongoose



DEMO 4 - VIEW

Template Engine

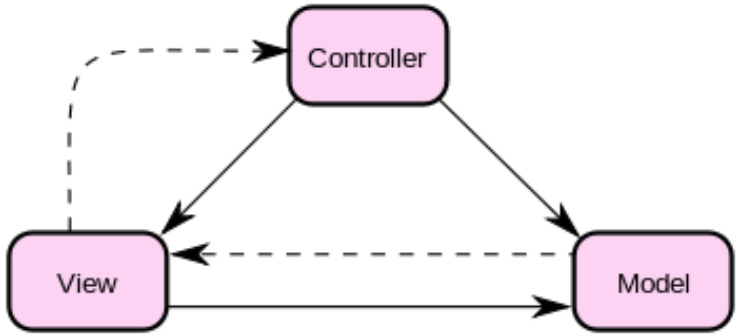
- **Ziel: Trennen von Controller und View.**
- **Express bietet eine render Methode an: app.render(view, [locals], callback)**
  - Die Render-Engine muss konfiguriert werden.

```
app.set('view engine', 'hbs');
```

  - Pfad der Templates:

```
app.set('views', __dirname + '/views');
```

- **Express wird mit Jade ausgeliefert**
  - http://jade-lang.com/
  - Vergleich: http://www.slant.co/topics/51/compare/~jade_vs_handlebars-js_vs_mustache-js
- **Handelbars nutzen**
  - npm install **express-hbs** --save



DEMO 5 - SESSION & SECURITY

Cookie

- **Details: https://github.com/expressjs/cookie-parser**
- **Installieren**
  - npm install cookie-parser –save
- **Einbinden**

```
app.use(require("cookie-parser")());
```

- **cookieParser(secret, options)**
  - secret: wird zum Signieren benötigt
- **Nutzen**
  - http://expressjs.com/api.html#res.cookie

Session

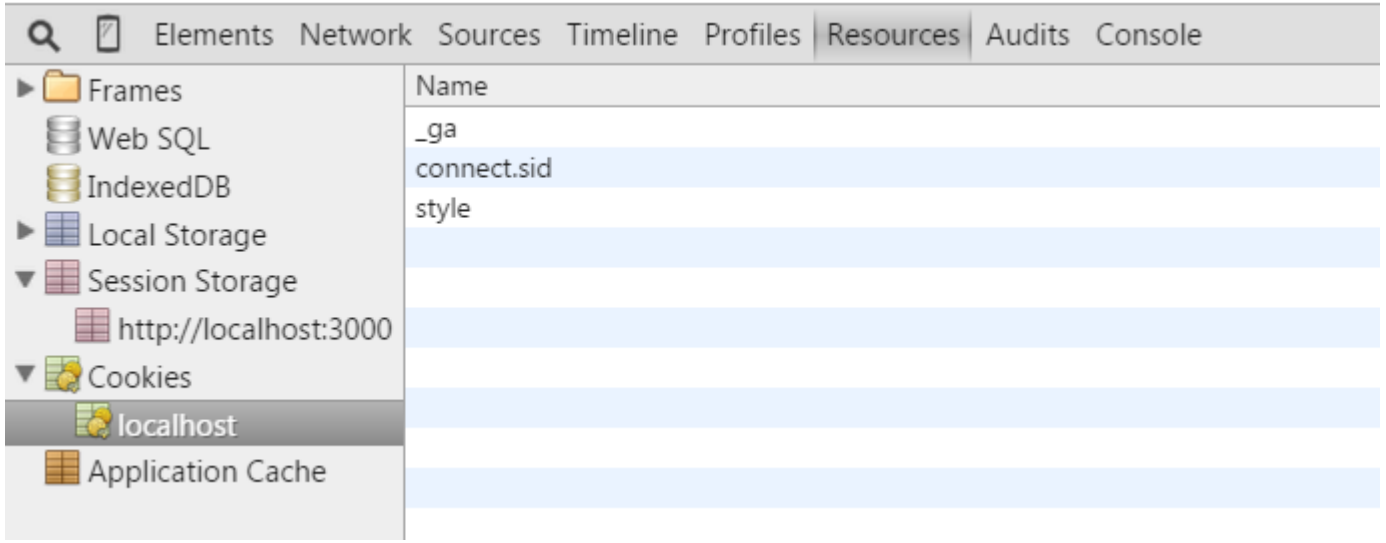
- **Session benötigt Cookies**
- **Installieren**
  - npm install session-parser –save
- **Einbinden**

```
app.use(session({ secret: ,'1234567', resave: false, saveUninitialized: true}));
```

- **Nutzen**
  - https://github.com/expressjs/session
- **Sonstiges**
  - Beim ersten «Connect» vom Client wird eine Session-Id erstellt und als Cookie zum Client geschickt.
  - Session-Data wird auf dem Server abgespeichert.

Cookie-Tools

- **Die meisten Developer Tools bieten die Möglichkeit gespeicherte Cookies anzuzeigen:**



Session

- **Was kann man nun damit machen?**
  - HTTP-Stateless umgehen
    - Wiederspricht REST!
    - Login-Status vom User abspeichern
    - *Allgemein*: Daten Server-seitig einem Benutzer zuordnen
  - Tracking

DEMO 6 - REST

REST

- **Was muss geändert werden?**

- **Rückgabeformat ändern**
- **Stateless**
  - Session durch ein Token ersetzten
- **…**

Unterschiedliche Formate

- **res.format : http://expressjs.com/api.html#res.format**

```
res.format({
    'text/html': function () {
        res.redirect("/");
    },
    'application/json': function () {
        res.send(true);
    },
});
```

Token

- **Ziel: Stateless Server**
- **Idee: Bei jeder Anfrage muss ein Token mitgeben werden.**
  - Im Token sind alle Informationen gespeichert, welche für eine Autorisierung notwendig sind.
    - Ausgestellt an
    - Ablauf-Datum
    - Signatur
  - Anhang von den Daten im Token kann der User wiederhergestellt werden.
- **Vorteil**
  - Jede Anfrage kann zu einem beliebigen Server gesendet werden; solange die Applikationen der Signatur vertraut.
- **Nachteile**
  - Was passiert wenn das Token «geklaut» wird?
  - Was passiert wenn das Smartphone geklaut wird?
    - Ablauf-Datum kurz setzten
    - Token invalideren

Token

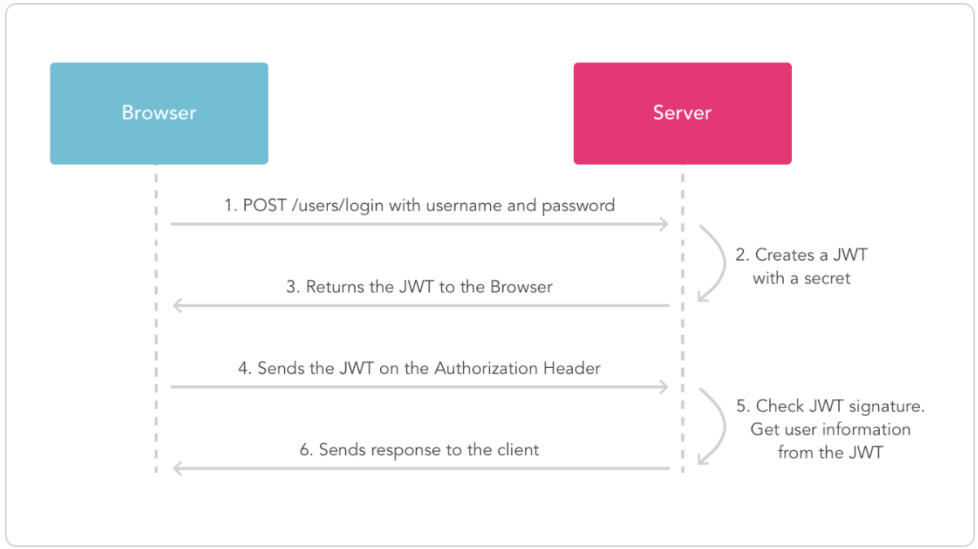
- **Wie erhält man das Token**
  - Token ist bekannt bzw. online sichtbar. Diese Variante wird oft bei REST APIs angeboten.
    - Amazon S3 Storage
    - search.ch: http://tel.search.ch/api/help
  - Bei einer Route müssen die persönlichen Daten angegeben werden. Falls korrekt, wird ein Token generiert und an den Client geschickt.

- **Libraries:**
  - http://passportjs.org/docs/oauth
  - https://github.com/auth0/express-jwt
  - https://github.com/auth0/node-jsonwebtoken
- **Beschreibung für jwt-tokens**
  - https://jwt.io/introduction/

JWT Token

***JSON Web Token** (**JWT**) is a JSON-based open standard (RFC 7519) for creating tokens that assert some number of claims.* *(Quelle: Wikipedia)*

- **HTTP-Header:** `Authorization: Bearer <token>`
- **Ablauf:**



- **Wichtig: Token nur über eine sichere Verbindung versenden.**

JWT Token

- **Beinhaltet Header, Payload (Claims), Signatur**

Header

Payload

Signatur

Encoded PASTE A TOKEN HERE

```
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1
bmlxdWVfbmFtZSI6WyJhZG1pbkBhZG1pbi5jaCIsI
mFkbWluQGFkbWluLmNoIl0sIm5hbWVpZCI6ImIxZj
BhZjZiLWRjOTctNGNmMi1iMTQ1LTdkZTRhNWFiZTN
kZiIsIm5iZiI6MTQ3MzQyMjQ3MiwiZXhwIjoxNDcz
NDIzMDcyLCJpYXQiOjE0NzM0MjI0NzIsImlzcyI6I
lBpenphIiwiYXVkIjoic2VsZiJ9.sdu0K62ozId5X
lC8CYTZ784uKr2Hp5JrQADzqMSOFieQEo1C4VwsYg
781hOmLgfm4Vq9XzPxEpCwgJS4wZtQxvJPIyX0KII
aWKIDS0vUxaQIsOPdcT9aqPOdaCbhVjSEWbn3AViW
57oXovQhWH8h4wcEH773N0WQef0klO6Z1C0eScrcZ
IPtsrfqlCHSHwV5ViC0WoRwQRcARoLj-
5238BMQfhiCWU1o6khPjJu469EscJPnAu1GGDi-
roFWdlnLeEp8c-
g4DECwNVmren1uyEHL_HaqZ2cLFPAVmP59p2PFEMO
```

Decoded EDIT THE PAYLOAD AND SECRET (ONLY HS256 SUPPORTED)

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "alg": "RS256",
  "typ": "JWT"
}
```

PAYLOAD: DATA

```
{
  "unique_name": [
    "admin@admin.ch",
    "admin@admin.ch"
  ],
  "nameid": "b1f0af6b-dc97-4cf2-b145-7de4a5abe3df",
  "nbf": 1473422472,
  "exp": 1473423072,
  "iat": 1473422472,
  "iss": "Pizza",
  "aud": "self"
}
```

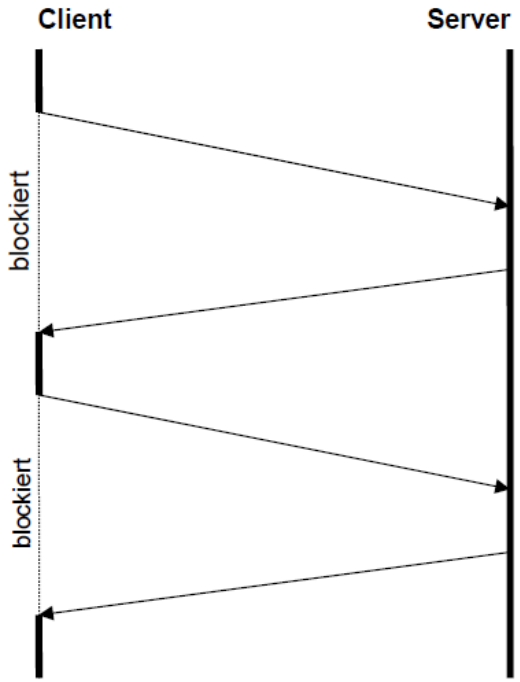
VERIFY SIGNATURE

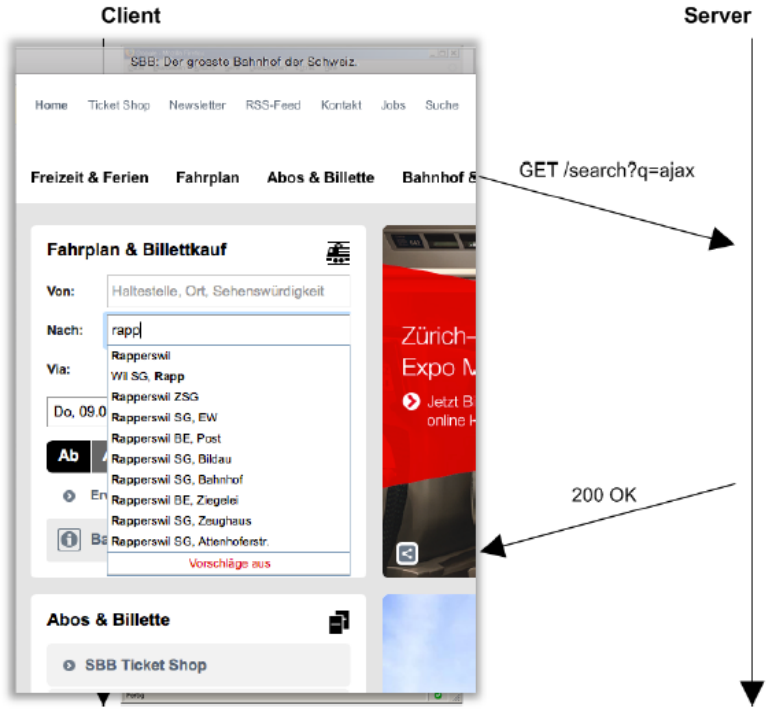
Tool: https://jwt.io/

DEMO 6 - AJAX

AJAX - Motivation

- **User Experience verbessern: Auto-Complete in Formularen**
  - Z.B. www.sbb.ch
- **Single Page Applikation**





AJAX - Definition

- **Asynchronous JavaScript and XML**
  - JSON hat sich durchgesetzt
- **Asynchronous**
  - Nachträgliches laden von Daten
  - Updaten von der Web Page

AJAX - Nachteile

- **Der State der Applikation ist nicht bekannt**
  - Schwer zu rekonstruieren
- **Suche-Maschinen führen kein JavaScript aus**
  - Seit 2014: Google führt JavaScript aus
- **Undo / Redo-Stack wird oft vergessen**
  - History kann angepasst werden
    - https://developer.mozilla.org/en-US/docs/Web/Guide/API/DOM/Manipulating_the_browser_history
  - Z.B. history.pushState({}, "Demo", "bar.html");
  - Sehr oft wird mit URL-Fragment dafür verwendet
    - history.pushState({}, "demo", window.location.pathname + "#demo3")
    - Führt zu weitern Problem
    - Siehe: https://developers.google.com/webmasters/ajax-crawling/docs/getting-started
  - Wird von Frameworks wie Angular und React übernommen!

XMLHttpRequest (*XHR*)

- **Für async Request erstellen zu können, wird XMLHttpRequest benötigt. Dies wird native von den Browsern zu Verfügung gestellt.**

```
var xhr = new XMLHttpRequest();
xhr.onreadystatechange = function() {
  if (4 === xhr.readyState) {
    if (xhr.status === 200) {
      alert(xhr.responseText);
    } else {
      alert('There was a problem with the request.');
    }
  }
};
xhr.open('GET', url, true); //async
xhr.send();
```

- **onreadystatechange wird aufgerufen falls sich der State ändert.**
- **https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest**

fetch

- **https://developer.mozilla.org/en/docs/Web/API/Fetch_API**
  - https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch
  - Neue Variante vom XMLHttpRequest
  - Promise
  - Polyfill vorhanden: https://github.com/github/fetch

```
fetch('/login', {
    method: 'POST',
    headers: {
        'Content-Type': 'application/json'
    },
    body: JSON.stringify({email: "admin@admin.ch", pwd: "123456"})
}).then(function (res) {
    console.log(res);
})
```

fetch

- **Cookies werden nicht automatisch mitgesendet:**
  - https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch#Sending_a_request_with_credentials_included

```
fetch('https://example.com', {
  credentials: 'include'
})
```

- **Fetch bringt einige Hilfsklassen mit um einen Request zu erstellen:**
  - https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch#Supplying_your_own_request_object

```
const myHeaders = new Headers();
myHeaders.append('Content-Type', 'text/plain');

const myInit = {
  method: 'GET',
  headers: myHeaders,
  cache: 'default'
};

const myRequest = new Request('/exampe', myInit);

fetch(myRequest) /*…*/
```

Einschub: SOP / CORS

- **Same-Origin-Policy**
  - Erlaubt XMLHttpRequest nur zur Origin.
  - https://de.wikipedia.org/wiki/Same-Origin-Policy
- **Cross-Origin Resource Sharing**
  - Mechanismus um Cross-Site-Requests zu ermöglichen.
  - Der Ziel-Server kann dem Client den Zugriff erlauben.
  - Wird vom Browser Enforced
- **Fehler falls CORS nicht aktiviert ist:**

```
XMLHttpRequest cannot load http://localhost:3000/data. No 'Access-Control-
Allow-Origin' header is present on the requested resource. Origin
'http://localhost:63342' is therefore not allowed access.
```

- **Wichtig: Chrome erlaubt kein CORS nach Localhost**

Einschub: SOP / CORS

- **CORS aktivieren**
  - http://enable-cors.org/
- **Express**
  - Module: https://github.com/expressjs/cors
  - Manuell:

```
const allowCrossDomain = function(req, res, next) {
    res.header('Access-Control-Allow-Origin', '*');
    res.header('Access-Control-Allow-Methods', 'GET,PUT,POST,DELETE');
    res.header('Access-Control-Allow-Headers', 'Content-Type');
    next();
};

app.use(allowCrossDomain);
```

WEBSOCKETS

Sockets

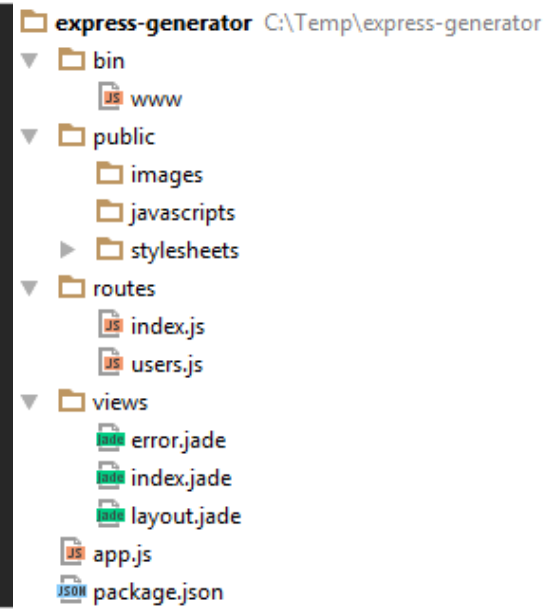
- **Das klassische Model vom Request-Response hat 2 Probleme**
  - Der Server kann keine Nachricht an den Client schicken
  - Jede Anfrage öffnete eine neue Verbindung
- **Dieses Model erschwert es real-time Apps zu machen z.B. Games, Chats.**
- **Eine Workaround war/ist Long-Polling**
  - Client öffnet eine Verbindung aber der Server beendet diese nicht.
  - Server sendet nun Nachrichten an den Client über diese Verbindung.
  - Problem
    - Timeouts (ca. 90s)
    - Offene Connections
- **WebSockets**
  - Ermöglicht «bi-directional», «always-on» Kommunikation
  - Komplexe API
    - http://socket.io/

GENERATOR

Generator

- **Link:** http://expressjs.com/en/starter/generator.html

```
$ express -h

  Usage: express [options] [dir]

  Options:

    -h, --help          output usage information
    -V, --version       output the version number
    -e, --ejs           add ejs engine support (defaults to jade)
        --hbs           add handlebars engine support
    -H, --hogan         add hogan.js engine support
    -c, --css <engine>  add stylesheet <engine> support (less|stylus|compass|sass) (defaults to plain css)
        --git           add .gitignore
    -f, --force         force on non-empty directory
```



- **Brauchbar als Start.**
  - Ordner für Service / Controller fehlt
  - ES6 Features fehlen z.B. kein const, let, arrows
- **Trennung zwischen Environment und Server-Bootstrapping**

DEPLOY

Cloud

- **Heruko:**
  - https://devcenter.heroku.com/articles/nodejs-support
  - https://devcenter.heroku.com/articles/getting-started-with-nodejs#introduction
- **Azure**
  - https://azure.microsoft.com/en-us/documentation/articles/app-service-web-nodejs-get-started/
- **…**
  - Überall wo Docker läuft
  - Überall wo Node.js läuft
- **Tipp fürs Projekt 2: (Für alle die keinen Server erstellen möchten)**
  - Firebase & Github Pages
    - REST Endpoint: https://firebase.google.com/
      - https://github.com/angular/angularfire2
        https://github.com/FirebaseExtended/reactfire#using-the-firebase-js-sdk-in-react
    - SPA-Container: https://help.github.com/articles/what-are-github-pages/

BONUS: GRAHQL

ÜBUNG

Aufgabe (falls Zeit)

Demo 2 verwendet aktuell einen Array zum Abspeichern der Daten. Speichern Sie die Daten in eine Datenbank z.B. mit nedb.

- **Demo klonen: https://github.com/gfeller/CAS_FEE_Expresss**
- **Installieren Sie https://github.com/louischatriot/nedb**
- **Ändern Sie orderStore**
  - Hinweis: nedb ist asynchrone
- **Ändern Sie den ordersController**
- **Testen Sie ihre Lösung**

Zeit: 20 Minuten

Die Lösung ist in Demo 3 integriert

Aufgabe (falls Zeit)

Demo 5 beinhaltet ein Login Dialog für die Authentifizierung. Die Autorisierung wurde bis jetzt ignoriert. Ändern Sie dies.

- **showOrder und deleteOrder darf nur vom selben User aufgerufen werden**
- **Testen Sie ihre Lösung**

Zeit: 15 Minuten
Die Lösung ist in Demo 6 integriert

Aufgabe (falls Zeit)

- **Analysieren Sie die Demos**
- **Überlegen Sie sich wie Sie Ihren Server gestalten.**
  - Datenbank
    - Nedb oder Mongodb (mit Mongoose?)
  - Data-Format wählen
  - …

Zeit: Bis zum Ende